

IBM WebSphere Message Broker XML Validator Plug-In Node

Version 1.1

7th July 2010

Mark Frost
WebSphere Message Broker Development
IBM United Kingdom Limited
Hursley Park
Winchester, UK

frostmar@uk.ibm.com

Property of IBM

Take Note!

Before using this report be sure to read the general information under "Notices".

Third Edition, July 2010

This edition applies to Version 1.1 of *IBM WebSphere Message Broker – XML Validator Plug-In* and to all subsequent releases and modifications unless otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2005, 2010.** All rights reserved. Note to US Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Table of Contents

Notices	4
Acknowledgments	5
Summary of Amendments	5
Preface	6
Possible Uses	6
Prerequisites	6
Installation	7
Uninstalling	10
What is an XML Schema	12
How the Node Works	14
Using the provided samples	15
Expected Behavior of Message Flows	17
Configuring the XML Validator Node	20
Specifying the Schema at run-time	22
Node Output	23

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- WebSphere
- WebSphere MQ (WMQ)
- WebSphere Business Integration Message Broker (WBI MB)

The following terms are trademarks of other companies:

- Windows 2000 Microsoft Corporation
- Linux Developed under the GNU General Public License

Acknowledgments

The XML Validator node is a Message Broker Java plug-in node that validates XML messages using the Apache Xerces parser. For more information on the Xerces parser and its capabilities, see <http://xml.apache.org/xerces2-j/>.

Summary of Amendments

Date	Changes
19 th April 2005	Initial release
28 th July 2006	Updated to support Message Broker v6
7 th July 2010	Updated to support Message Broker v7

Preface

This SupportPac supplies a Message Broker plug-in node that validates an XML message against an XML schema. If the message conforms to the schema, the message is passed to the Out terminal unchanged. However, if errors are encountered these are added to the Environment tree, and passed to the Invalid terminal, along with the unchanged message.

The XML Validator node checks the bitstream of a message, as opposed to the logical structure, and therefore supports any message format, e.g. XML, XMLNS, BLOB etc.. Configurable parameters are supplied to allow the schema defined in the XML document to be overridden, and can be set either via the node properties or at runtime, by the use of Environment/LocalEnvironment variables.

The node is implemented as a Java plug-in, and therefore supports all platforms where Java plug-ins may be used.

Possible uses

This SupportPac could be used as part of a Web Services message flow, if there is a requirement to validate incoming, or outgoing, XML messages against schemas. The ability to validate against a different schema for each message enables the node to be used in situations where there are a large variety of messages passing through the flow. For instance, if the broker is acting as a Web Services intermediary, several types of messages could be validated by one message flow using this node.

It is also useful when the content of a schema changes frequently, such as during application development stages, as no re-deploy of the message flow is required to use the changed schema.

The plug-in node could also be used to check messages against schemas for message flows that only use the XML or XMLNS domains.

Prerequisites

This SupportPac requires IBM WBI Message Broker version 5.0 or above, and has been developed on the v5.0 FixPack 4 level of the product on both Microsoft Windows and Linux. Additional testing has been performed with Message Broker v6.0 on Microsoft Windows, Linux and AIX; and Message Broker v7.0 on Microsoft Windows. Message Broker running on Z/OS is not supported.

Installation

The plug-in node consists of two parts, a run-time JAR file (*XMLValidator.jar*), and a design-time Toolkit plug-in (*com.ibm.mq.supportpacs.XMLValidator*) which provides the node for use in message flows.

Installing the run-time component

After unzipping the SupportPac zip file, copy the *runtime_all_versions/XMLValidator.jar* file to all of the machines running brokers that are required to run the node. The JAR file should be placed in the *<WBI MB Install Directory>/jplugin* location, where *<WBI MB Install Directory>* is the where the WBI Message Broker product was installed. This defaults to:

Message Broker v5:

- *C:\Program Files\IBM\WebSphere Business Integration Message Brokers* on Microsoft Windows
- */opt/wmqi/* on Linux

Message Broker v6:

- *C:\Program Files\IBMMQS\6.0* on Microsoft Windows
- */opt/ibm/mqsi/6.0/* on Linux

Message Broker v7:

- *C:\Program Files\IBMMQS\7.0* on Microsoft Windows
- */opt/ibm/mqsi/7.0/* on Linux


Alternatively, the JAR file can be copied to any other directory readable by the broker, as long as that directory is added to the broker's *lil* path (use command *mqsichangebroker* to set the *lil* path).

Ensure that the JAR file is readable by the user used to run the broker. After copying the JAR file the brokers need to be restarted before any message flows that use the node can be deployed.

Installing the design-time component

Message Broker v5

The SupportPac uses the Install/Update perspective of the Message Broker Toolkit v5 to install the design-time component.

1. After unzipping the SupportPac zip file to a temporary location, launch the Message Brokers Toolkit, and open the *Install/Update* perspective. The *Install/Update* perspective can be opened using the menu option *Window → Open Perspective... → Other...*, selecting the *Install/Update* option, and clicking OK.
2. In the Feature Updates view, open the *My Computer* section, and navigate to *</A9A>/tooling_v5/install*. The directory containing the SupportPac should have a different icon () to the other directories, and may appear at the bottom of the directory list.
3. Expand the elements beneath the directory by clicking on the “+” icons, and select the “*WebSphere Business Integration Message Broker XML Validation Plug-in Node Feature 1.0.0*” entry.
4. In the preview pane (usually the largest view open), click the *Install* button to begin the installation.
5. The install wizard starts up, select the appropriate responses to the questions, and restart the Toolkit to use the node. The Toolkit may warn that the feature is unsigned, this is normal.

After a successful installation, the node is available in the message flow node palette, under the section *XML Validator*, and when placed in a message flow, looks like this:



Help content is also installed in the *Help → Help Contents... → WebSphere Business Integration Message Broker → Reference → Message Flows → User Defined nodes* section.

Message Broker v6

The SupportPac uses the *Software Updates* function of the Message Broker Toolkit v6 to install the design-time component.

1. After unzipping the SupportPac zip file to a temporary location, launch the Message Broker Toolkit, and use the *Software Update* tool to search for new features. The *Software Update* can be opened using the menu option *Help → Software Updates → Find and Install...*

2. In the Install/Update dialog, select *Search for new features to install* and select *Next*.
3. Add a *New Local Site...* browsing to the location *<IA9A>/tooling_v6/update_site* and select *OK*
4. Back in the *Install* dialog, you should now be able to expand the tree under your new location (click the '+' icon), and select the update "*IBM WebSphere Business Integration Message Broker XML Validator Node*". Click *Next*
5. Check the feature listed "*WebSphere Business Integration Message Broker XML Validator Plug-in Node Feature*" and select *Next*
6. Accept the license agreement and select *Next*
7. Select a site to contain the new plug-in – any site is acceptable – and select *Finish*. Restart the Toolkit to use the new node. The Toolkit may warn that the feature is unsigned, this is normal.

Message Broker v7

The SupportPac uses the *Software Updates* function of the Message Broker Toolkit v7 to install the design-time component.

1. After unzipping the SupportPac zip file to a temporary location, launch the Message Broker Toolkit, and use the *Software Update* tool to search for new features. The *Software Update* can be opened using the menu option *Help → Software Updates...*
2. In the *Available Software* tab of the dialog select *Add site*
3. Select *local* and navigate to the *<IA9A>/tooling_v7/update_site* directory, press [ok]
eg *file:/C:\ia9a\tooling_v7\update_site*
4. A new item "com.ibm.mq.supportpacs.feature.XMLValidator Package" should be displayed. Tick its checkbox and click *Install...*
5. After some processing select *Next* on page "Review and confirm that the checked items will be installed"
6. Accept the licence and select *Finish*. Restart the Toolkit to use the new node.

Uninstalling

Message Broker v5

1. Open the *Install/Update* perspective using the menu option *Window → Open Perspective... → Other...*, selecting the *Install/Update* option, and clicking *OK*.
2. In the *Install Configuration* view, open the *Current Configuration* entry. Open the entry "*file:c:/Program Files/IBM/WebSphere Business Integration Message Brokers/eclipse*", and select the *XML Validator* feature entry.
3. In the *Preview* pane, click the *disable* button. This prevents the use of the *XMLValidator* node within the *Toolkit*, but leaves the files on the disk. The feature may be enabled at a later date if required. To completely remove it, delete the following folders from the *Message Broker* install directory ("*<WBI MB Install Directory>*", as mentioned in the *Install* section):
 - eclipse\plugins\com.ibm.mq.supportpacs.XMLValidator_1.0.0*
 - eclipse\features\com.ibm.mq.supportpacs.feature.XMLValidator_1.0.0*
4. The runtime component should also be removed from all broker machines it was copied to. This is done by stopping the brokers, deleting the *XMLValidator.jar* file from the *jplugin* directory in the *Message Broker* install directory, and restarting the brokers on the machine.

Message Broker v6

1. To disable or remove the *XML Validator* node, use *Help → Software Updates → Manage Configuration...*
2. In the *Installed Software* tab of the dialog find and select "*WebSphere Business Integration Message Broker XML Validator Plug-in Node Feature*" from the tree on the left. It will be listed under the site the plug-in was installed to.
3. Choose *Disable* to prevent the node from being used, or *Uninstall* to also delete the node files from disk
4. The runtime component should also be removed from all broker machines it was copied to. This is done by stopping the brokers, deleting the *XMLValidator.jar* file from the *jplugin* directory in the *Message Broker* installation, and restarting the brokers.

Message Broker v7

1. To uninstall the XML Validator node, use *Help* → *Software Updates...*
2. On the *Installed Software* tab, find and select *com.ibm.mq.supportpacs.feature.XMLValidator Package* from the list. Choose *Uninstall* to delete the node files from disk
3. The runtime component should also be removed from all broker machines it was copied to. This is done by stopping the brokers, deleting the *XMLValidator.jar* file from the *jplugin* directory in the Message Broker installation, and restarting the brokers.

What is an XML Schema?

An XML Schema is a document that defines the format of an XML document. It outlines what XML elements are allowed, how many elements can be present, and in what order. It also allows the content of the elements to be specified.

If an XML instance document contains only XML elements and data that match the definitions present in the XML schema, it is said to conform to the schema.

Namespaces may also be used to prevent clashes of element names and definitions.

In the example below, the schema specifies that there is one element called “AnElement” which may contain the values A, B, or C. It also defines the namespace in which the elements reside (“<http://www.ibm.com>”), and a shortcut for that namespace (“AXMLSchema”).

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://
www.ibm.com" xmlns:AXMLSchema="http://www.ibm.com">
  <element name="AnElement">
    <simpleType>
      <restriction base="string">
        <enumeration value="A"></enumeration>
        <enumeration value="B"></enumeration>
        <enumeration value="C"></enumeration>
      </restriction>
    </simpleType>
  </element>
</schema>
```

The instance document below specifies the namespace of the documents, and the schema file it is based upon. It then continues to define the element “AnElement” and value of “A”. It is therefore a valid XML document, and in the case of the XMLValidator node, would be propagated to the Out terminal. If any other elements were defined, or the value was not A, B, or C, or if the namespace were different, it would be an invalid document. If this document were passed through the XML Validator node, it would be sent to the Invalid terminal.

```
<?xml version="1.0" encoding="UTF-8"?>
<AXMLSchema:AnElement xmlns:AXMLSchema="http://www.ibm.com" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.ibm.com
```

```
NewXMLSchema.xsd ">  
A</AXMLSchema:AnElement>
```

XML Schema documents often have a file extension of “xsd”.

For more detail on XML Schemas, see the World Wide Web Consortium web site: <http://www.w3.org/XML/Schema>

How The Node Works

The XML Validator node utilizes the Apache Xerces XML parser (see <http://xml.apache.org/xerces2-j/>). When a message flow is deployed, the configurable parameters set in the message flow are passed to the node. The node does not interact with other instances of the node deployed in the same message flow, or to the same broker.

When a message arrives at the input terminal, a DOMParser object is created, and a custom ErrorHandler is registered with the parser, which is used to store all errors encountered by the parser. The bitstream of the message body (i.e. the last child of the root element) is taken and passed to the parser object via the parse() method. Upon completion, the ErrorHandler is checked to see if it contains any errors, if it does the message and errors are passed to the Invalid terminal, however, if the message conforms to the schema, it is propagated down the Out terminal.

For each message that passes through the node, a new parser object is created. This means that each message requires the parser to read the schema, which may be a time consuming task if the schema is located remotely. By placing the schema locally the read time will be reduced.

The source for the plug-in is provided in the “src” directory where the SupportPac zip file was unzipped.

Using the provided samples

There are three sample flows included, each demonstrating the different methods of specifying an XML schema.

1. Using the message's *xsi:schemaLocation* attribute. (ValidateAccordingToMsg.msgflow)
2. Using the node's properties. (ValidateAccordingToNode.msgflow)
3. Using the message flow environment variables. (ValidateAccordingToVariables.msgflow)

The *Invalid* terminals for each of the XML Validator nodes is connected to a compute node that constructs an error message containing the parser errors from the Environment tree, and a copy of the message.

Trace nodes are included in the message flows, to trace incoming messages, and failed messages along with errors. To see the output from these nodes, enable and capture User Trace from the message flow (see the *Using Trace* section of the *Diagnosing errors* help category)

The *samples* directory contains sample message flows, messages and schemas. The samples should be imported into the workspace using the *File* → *Import* → *Existing Project Into Workspace* tool, and selecting the *XMLValidatorSampleFlows* directory within the *samples* directory (in the location the SupportPac was unzipped to).

The *defqs.mqsc* file in the imported project contains the queue definitions used by the message flows. It should be applied to the broker's queue manager before deploying the message flow using the *runmqsc <QMGR> <defqs.mqsc* command.

Copy the sample messages and schemas from the *samples\XMLValidatorSampleFlows\SampleMessages* and *samples\XMLValidatorSampleFlows\SampleSchemas* to the *c:* directory, if possible. If the broker is running on an different machine, use standard file transfer tools to copy the schemas to the remote machine.

Edit the sample message flows and messages, if the schema files are not located in the *c:* directory. The default settings of the samples expect to find the schemas in the *c:* directory of the broker machine. If the samples are not being run on Windows, or the schemas are located in a different directory, change the *file://* references,

e.g. If the schemas are found in the */home/user* directory,

```
xsi:schemaLocation="file://MyNameSpace file:///c:/CustomerOrder.xsd"
```

becomes:

```
xsi:schemaLocation="file://MyNameSpace file:///home/user/CustomerOrder.xsd"
```

(The namespace "*file://MyNameSpace*" remains unchanged)

The following values will need modifying:

1. For the *ValidateAccordingToNode* message flow, the *Schema Location* and *No Namespace Schema Location* XMLValidator node attributes in the message flow.
2. For the *ValidateAccordingToMsg* message flow, the *xsi:schemaLocation* and *xsi:noNamespaceSchemaLocation* attributes in the sample messages..
3. For the *ValidateAccordingToVars* message flow, the *XMLValidator_SetVars* ESQL code will need updating.

Once the message flows have been deployed to the broker, the messages can be sent to the input queues using standard tools, such as those found in SupportPac IH03.

Expected Behavior of Message Flows:**ValidateAccordingToMsg Message Flow**

Input queue: XMLVALID.MSG.IN

Input file: CustomerOrder.xml

This is a valid XML instance document, and should be passed through to the XMLVALID.MSG.OUT queue untouched.

Things to try: Rename the CustomerOrder.xsd file, and resend the message. As the schema can no longer be found, the message is considered to be invalid, and will be propagated to the XMLVALID.MSG.INVALID queue. The compute node places the errors from the parser, along with the message, e.g.:

```
<MessageFailed>
  <Errors>
    <Error>Warning: [2:176] schema_reference.4: Failed to read schema document
    &apos;file:///c:/CustomerOrder.xsd&apos;; because 1) could not find the
    document; 2) the document could not be read; 3) the root element of the document
    is not &lt;xsd:schema&gt;.</Error>
    <Error>Error: [2:176] cvc-elt.1: Cannot find the declaration of element
    &apos;Message:Order&apos;.</Error>
    <Error>Warning: [3:29] schema_reference.4: Failed to read schema document
    &apos;file:///c:/CustomerOrder.xsd&apos;; because 1) could not find the
    document; 2) the document could not be read; 3) the root element of the document
    is not &lt;xsd:schema&gt;.</Error>
    <Error>Warning: [17:19] schema_reference.4: Failed to read schema document
    &apos;file:///c:/CustomerOrder.xsd&apos;; because 1) could not find the
    document; 2) the document could not be read; 3) the root element of the document
    is not &lt;xsd:schema&gt;.</Error>
  </Errors>
  <Message>
    <UnknownParserName></UnknownParserName>
    <BLOB> ... Message ... </BLOB>
  </Message>
</MessageFailed>
```

Rename the CustomerOrder.xsd file back to the original name.

Input file: CustomerOrder_invalid.xml

This is an invalid XML instance document, the CustomerID field is too short, and the price is too large. The message is therefore propagated to the XMLVALID.MSG.INVALID queue, with messages indicating the errors.

Input file: CustomerOrderNoNS.xml

This is a valid XML instance document, which does not use namespaces. This should be propagated to the XMLVALID.MSG.OUT queue.

Input file: CustomerOrderNoNS_invalid.xml

This is an invalid XML instance document, the title field value is not present in the enumeration, and the price is too large. The message is propagated to the XMLVALID.MSG.INVALID.

ValidateAccordingToNode / ValidateAccordingToVariables Message Flows

These flows are quite similar, they should ignore the schema location values specified in the message, and use the values specified in the message flow.

Input Queue: XMLVALID.NODE.IN / XMLVALID.ENV.IN

Input File: CustomerOrder.xml

This is a valid message, so should propagate to the XMLVALID.NODE.OUT or XMLVALID.ENV.OUT

Things to try: Change the value of the xsi:schemaLocation attribute found in the CustomerOrder.xml file, and resend. It should still be considered a valid message as the message flow overrides the invalid values. (You can confirm the message is invalid by passing through the ValidateAccordingToMsg flow).

Input File: CustomerOrderNoNS.xml

This is a valid message, with no namespace, and should propagate to the XMLVALID.NODE.OUT or XMLVALID.ENV.OUT.

Things to try: Modify the `xsi:noNamespaceSchemaLocation` attribute, and resend. The message should still be considered valid, as the attribute is being overridden as before.

Input File: CustomerOrder_invalid.xml, CustomerOrderNoNS_invalid.xml

These two messages are invalid, and should be propagated to the XMLVALID.NODE.INVALID / XMLVALID.ENV.INVALID queue.

Configuring the XML Validator Node

After adding the XML Validator node to a message flow, it may be configured by right-clicking on the node and selecting the Properties... option.

Basic Parameters

xsi:schemaLocation

This maps to the Xerces property "<http://apache.org/xml/properties/schema/external-schemaLocation>" (see <http://xml.apache.org/xerces2-j/properties.html>)

Forces the parser to override the *xsi:schemaLocation*¹ attribute in the XML document, and use the value supplied. This property can also be used to set the schema if the attribute is not present in the document. The value is the same syntax as an *xsi:schemaLocation* attribute, and may be a list of namespace and schema document location pairs, separated by a space, e.g.:

```
namespace1 file:///c:/xsd1.xsd namespace2 http://www.ibm.com/xsd2.xsd
```

To specify a schema on the file system, the file must be available to the broker, and specified using the URI prefix *file:///c:/*, or *file:///home/user/* for example. It is also possible to use the URI prefix *http://* to reference remote schemas that are accessible via the http protocol from the broker.

xsi:noNamespaceSchemaLocation

This maps to the Xerces property "<http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation>" (see <http://xml.apache.org/xerces2-j/properties.html>)

This forces the parser to override the value of *xsi:noNamespaceSchemaLocation*² found in an XML document, and use the schema specified. This property can also be used to set the schema if the attribute is not present in the document. Only one value can be provided, and is in the form of a filename.

To specify a schema on the file system, the file must be available to the broker, and specified using the URI prefix *file:///c:/*, or *file:///home/user/* for example. It is also possible to use the URI prefix *http://* to reference remote schemas that are accessible via the http protocol from the broker.

1 see http://www.w3.org/TR/xmlschema-1/#xsi_schemaLocation

2 see http://www.w3.org/TR/xmlschema-1/#xsi_schemaLocation

Ignore Warnings

If this parameter is set, any warnings reported by Xerces will be ignored by the node, and will not cause the message to be considered invalid. If it is not set, warnings will be reported in the Environment tree, and if a warning is encountered, the message will be propagated to the Invalid terminal.

Full Schema Check

Tells the parser to perform full grammar constraint checking of the schema itself, which may be time consuming. This does not affect the level of checking of the XML document, but may find errors in the schema that normally would not be found. This may be useful during test and development phases to check the validity of the schema.

Advanced Parameters

Enable Logging

A check box to enable or disable logging output from the node. The Log filename parameter determines where the log is output to. Logging should only be used to debug, as the log file will not wrap, and therefore could grow very large.

Log Filename

Enables the log output to be redirect to a file. If left blank, the log is output to the standard output. To view the broker's standard output on the Microsoft Windows operating system, open the services control panel applet, and modify the broker service to Log On as the system account, and enable it to interact with the desktop. On Unix operating systems, redirect to a file, and use the operating system utility "tail" to view the output as it is updated.

If several XML Validator nodes are present on a broker, specify a different filename for each, so that the output from each node is distinguishable.

Specifying the Schema at run-time

If no schema properties are set, the schema specified in the XML document is used to validate the document. The schema values can be overridden by either specifying values in the message flow, or by setting message flow variables as a message is being processed. The different methods are outlined in the following list, in order of precedence.

Schema Location

- Message flow node settings
- `LocalEnvironment.XMLValidator.SchemaLocation`
- `Environment.XMLValidator.SchemaLocation`

NoNamespace Schema Location

- Message flow node settings
- `LocalEnvironment.XMLValidator.NoNamespaceSchemaLocation`
- `Environment.XMLValidator.NoNamespaceSchemaLocation`

Node Output

If there are no validation errors the message is passed unchanged to the Out terminal. If errors do occur, the Root tree of the message is passed unchanged to the Invalid terminal, and errors are reported in the Environment tree, in the `Environment.XMLValidationErrors.Error[]` location. Multiple Error elements may be returned, if more than one error is found.

An example of an error returned by the parser is:

```
Error: [5:24] cvc-enumeration-valid: Value 'Ms' is not facet-valid with respect to enumeration '[Mr, Mrs, Miss, Dr]'.
```

The first word of the error text determines whether the failure is an error or a warning. The second part gives a location of the error, (in this instance, line number 5). The final part outlines what the problem was.

END OF DOCUMENT